**CODING PROBLEMS (12.11.2024)**

1. **ANAGRAM**

class Solution {

public static boolean areAnagrams(String s1, String s2) {

HashMap<Character,Integer> hm = new HashMap<>();

for(char c : s1.toCharArray()){

hm.put(c, hm.getOrDefault(c,0)+1);

}

for(char c: s2.toCharArray()){

hm.put(c, hm.getOrDefault(c,0)-1);

}

for(var n: hm.entrySet()){

if(n.getValue() !=0){

return false;

}

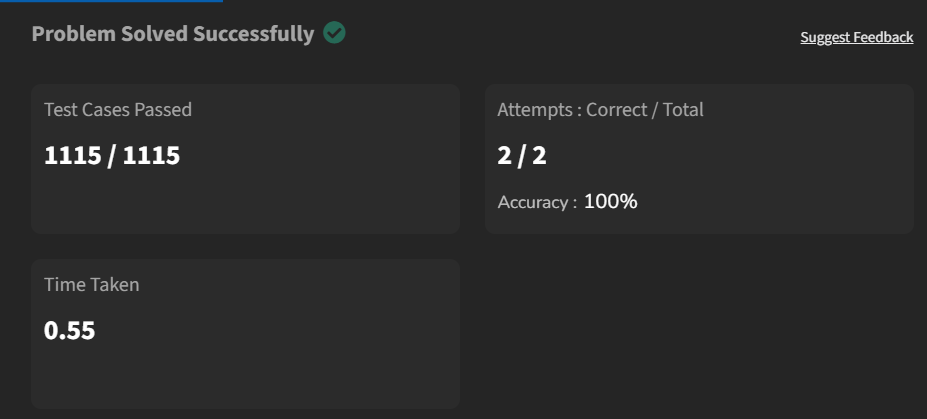
}

return true;

}

}

OUTPUT:



Time Complexity : O(N)

Space Complexity : O(N)

1. **Row with Maximum 1’s**

class Sol

{

public static int maxOnes (int mat[][], int N, int M)

{

int R = mat.length;

int maxRow = -1;

int row = 0;

int column = mat[0].length-1;

while(row<R && column>=0){

if(mat[row][column]==0){

row++;

}

else{

maxRow = row;

column--;

}

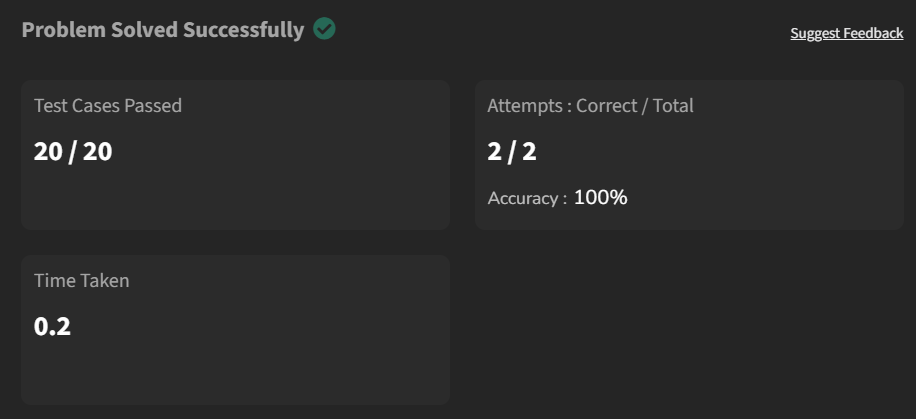
}

return maxRow;

}

}

OUTPUT:



Time Complexity : O(logN)

Space Complexity : O(1)

1. **Longest Consecutive Subsequence**

class Solution {

public int findLongestConseqSubseq(int[] arr) {

Arrays.sort(arr);

int count = 1;

int max = 1;

for(int i=1; i<arr.length; i++){

if(arr[i]-arr[i-1]==1){

count+=1;

}

else if(arr[i]-arr[i-1]==0){

}

else{

count = 1;

}

max = Math.max(count,max);

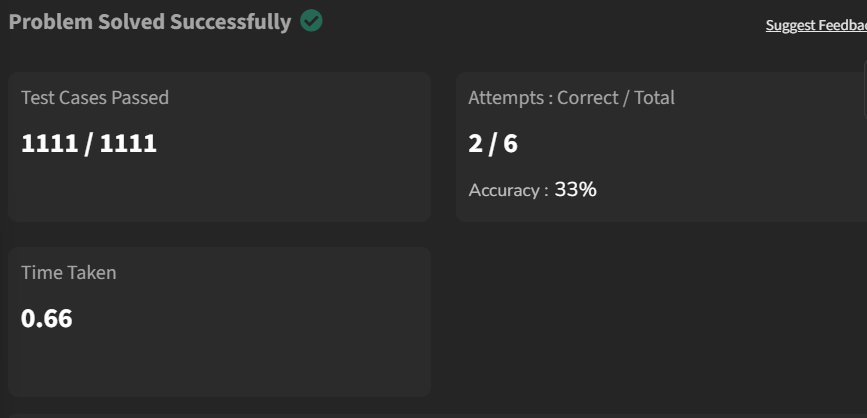
}

return max;

}

}

OUTPUT:



Time Complexity : O(NlogN)

Space Complexity : O(1)

1. **Longest Palindrome in a String**

class Solution {

static String longestPalindrome(String s) {

if (s.length() <= 1) {

return s;

}

int maxLen = 1;

int start = 0;

int end = 0;

boolean[][] dp = new boolean[s.length()][s.length()];

for (int i = 0; i < s.length(); ++i) {

dp[i][i] = true;

for (int j = 0; j < i; ++j) {

if (s.charAt(j) == s.charAt(i) && (i - j <= 2 || dp[j + 1][i - 1])) {

dp[j][i] = true;

if (i - j + 1 > maxLen) {

maxLen = i - j + 1;

start = j;

end = i;

}

}

}

}

return s.substring(start, end + 1);

}

}

OUTPUT:



Time Complexity : O(N^2)

Space Complexity : O(N^2)

1. **Rat in a Maze**

class Solution {

public ArrayList<String> findPath(int[][] mat) {

StringBuilder str = new StringBuilder();

ArrayList<String> ans = new ArrayList<>();

find(mat, str, ans, 0 , 0);

return ans;

}

public void find(int[][] mat, StringBuilder str, List<String> ans, int i, int j) {

if(i<0 || j<0 || i>=mat.length || j>=mat.length || mat[i][j]==0) return;

if(i==mat.length-1 && j==mat.length-1) {

ans.add(str.toString());

}

mat[i][j] = 0;

str.append('U');

find(mat, str, ans, i-1, j);

str.deleteCharAt(str.length()-1);

str.append('R');

find(mat, str, ans, i, j+1);

str.deleteCharAt(str.length()-1);

str.append('D');

find(mat, str, ans, i+1, j);

str.deleteCharAt(str.length()-1);

str.append('L');

find(mat, str, ans, i, j-1);

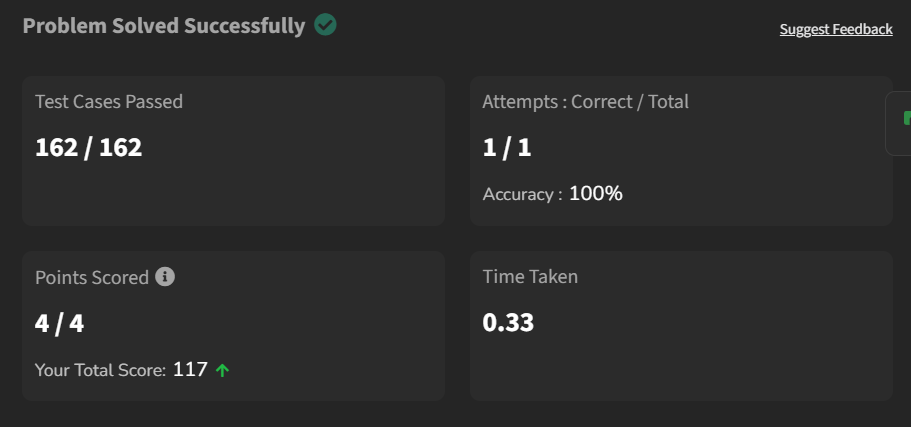
str.deleteCharAt(str.length()-1);

mat[i][j] = 1;

}

}

OUTPUT:



Time Complexity : O(N^2)

Space Complexity : O(N^2+P)